برای تهیه یک فایل ورد از اطلاعات ارائه‌شده در مکالمه، محتوای مرتبط با پرس‌وجوهای شما و پاسخ‌های داده‌شده را سازمان‌دهی کرده و به صورت یک سند متنی ساختارمند ارائه می‌کنم. این سند شامل بخش‌های زیر خواهد بود:

مقدمه: توضیح هدف و زمینه پرس‌وجوها.

تحلیل مشکل ویزارد فاکتور: توضیح مشکل دکمه «مرحله بعد» در ویزارد و راه‌حل‌های پیشنهادی.

طراحی ساختار سازمانی: توضیح مدل‌های پیشنهادی برای سازمان، معاونت‌ها، ادارات، شعب، و نقش‌های تأیید/رد کننده.

کدهای پیشنهادی: شامل مدل‌ها، فرم‌ها، مهاجرت‌ها، و منطق ویوها.

ملاحظات و نکات: نکات مربوط به مهاجرت داده‌ها، عملکرد، و رابط کاربری.

از آنجا که نمی‌توانم مستقیماً فایل ورد (.docx) را در این محیط تولید کنم، متن کامل را به صورت ساختارمند ارائه می‌دهم که می‌توانید آن را کپی کرده و در یک ویرایشگر متن مانند Microsoft Word جای‌گذاری کنید. همچنین، اگر نیاز به فایل ورد دارید، می‌توانید متن را در Word کپی کنید یا از ابزارهای تبدیل آنلاین برای تولید فایل .docx استفاده کنید.

سند تحلیل و طراحی سیستم سازمانی و ویزارد فاکتور

مقدمه

این سند به تحلیل دو پرس‌وجوی اصلی مرتبط با یک سیستم مدیریت سازمانی و فاکتور در یک پروژه جنگو می‌پردازد:

مشکل ویزارد فاکتور: دکمه «مرحله بعد» در مرحله آیتم‌های فاکتور (factor\_items) کار نمی‌کند و ویزارد به مرحله بعدی نمی‌رود.

طراحی ساختار سازمانی: نیاز به پیاده‌سازی یک ساختار سلسله‌مراتبی شامل سازمان، معاونت‌ها، ادارات، و شعب، با امکان تعریف نقش‌های تأیید/رد کننده برای پست‌ها.

هدف این سند ارائه تحلیل مشکلات، راه‌حل‌های پیشنهادی، و کدهای لازم برای پیاده‌سازی است. این اطلاعات برای استفاده بعدی و توسعه سیستم نگه‌داری می‌شود.

تحلیل مشکل ویزارد فاکتور

شرح مشکل

در ویزارد ایجاد فاکتور (FactorCreateWizardView)، دکمه «مرحله بعد» در مرحله factor\_items (آیتم‌های فاکتور) کار نمی‌کند. لاگ‌ها نشان می‌دهند که درخواست POST با کد وضعیت 200 پاسخ داده شده، اما به جای ریدایرکت به مرحله بعدی (factor\_docs)، صفحه فعلی دوباره رندر می‌شود. این رفتار معمولاً به دلیل نامعتبر بودن فرم‌ست یا مشکل در ذخیره داده‌ها در storage رخ می‌دهد.

جزئیات لاگ‌ها

مرحله فعلی: factor\_items (فرم‌ست آیتم‌های فاکتور).

داده‌های POST:

یک ردیف معتبر با فیلدهای description, quantity, و unit\_price.

ردیف دوم خالی و برای حذف علامت‌گذاری شده (DELETE=True).

پاسخ HTTP: کد 200 (بدون ریدایرکت).

مشکل احتمالی: فرم‌ست معتبر تشخیص داده نشده یا داده‌ها به درستی در storage ذخیره نمی‌شوند.

علل احتمالی

اعتبارسنجی فرم‌ست: ممکن است فرم‌ست به دلیل عدم رعایت شرط min\_num=1 یا خطای پنهان نامعتبر باشد.

تنظیمات instance: متد get\_form\_instance برای factor\_items مقدار None برمی‌گرداند، که ممکن است باعث مشکل در اتصال فرم‌ست به مدل Factor شود.

ذخیره داده‌ها: داده‌های فرم‌ست ممکن است به درستی در storage ذخیره نشوند.

قالب یا جاوااسکریپت: ممکن است خطایی در قالب HTML یا جاوااسکریپت مانع ارسال صحیح داده‌ها شود.

راه‌حل‌های پیشنهادی

برای رفع مشکل، تغییرات زیر پیشنهاد شد:

۱. بهبود لاگ‌گیری در متد post

متد post در FactorCreateWizardView اصلاح شد تا خطاهای فرم‌ست را به دقت لاگ کند و وضعیت اعتبارسنجی را بررسی کند:

python

def post(self, \*args, \*\*kwargs):

logger.info(f"[ویزارد-POST] شروع اجرای متد post بازنویسی شده. مرحله فعلی از storage: {self.steps.current}")

logger.debug(f"[ویزارد-POST] داده‌های POST: {self.request.POST}")

logger.debug(f"[ویزارد-POST] فایل‌های FILES: {self.request.FILES}")

from formtools.wizard.forms import ManagementForm

management\_form = ManagementForm(self.request.POST, prefix=self.prefix)

if not management\_form.is\_valid():

logger.error("[ویزارد-POST] management\_form نامعتبر است! خطاها: %s", management\_form.errors.as\_json())

messages.error(self.request, \_("خطای داخلی: اطلاعات کنترل مراحل ویزارد نامعتبر است."))

try:

current\_step\_on\_error = self.request.POST.get(self.management\_form.add\_prefix('current\_step'), self.steps.first)

form\_on\_error = self.get\_form(step=current\_step\_on\_error, data=self.request.POST, files=self.request.FILES)

return self.render(form\_on\_error)

except Exception:

return self.render\_goto\_step(self.steps.first)

current\_step = management\_form.cleaned\_data["current\_step"]

logger.info(f"[ویزارد-POST] مرحله شناسایی شده از management\_form: '{current\_step}'")

form = self.get\_form(step=current\_step, data=self.request.POST, files=self.request.FILES)

from django.forms.formsets import BaseFormSet

if isinstance(form, BaseFormSet):

if form.is\_valid():

logger.info("[ویزارد-POST] فرم‌ست مرحله ('%s') معتبر است.", current\_step)

valid\_forms = [f for f in form.forms if f.is\_valid() and not f.cleaned\_data.get('DELETE', False)]

logger.debug(f"[ویزارد-POST] تعداد فرم‌های معتبر: {len(valid\_forms)}")

else:

formset\_errors = []

for i, subform in enumerate(form.forms):

if subform.errors:

formset\_errors.append({f"فرم {i}": subform.errors})

logger.warning("[ویزارد-POST] فرم‌ست مرحله ('%s') نامعتبر است. خطاها: %s", current\_step, json.dumps(formset\_errors, ensure\_ascii=False))

logger.debug("[ویزارد-POST] داده‌های فرم‌ست: %s", [f.cleaned\_data for f in form.forms if f.cleaned\_data])

else:

if form.is\_valid():

logger.info("[ویزارد-POST] فرم مرحله ('%s') معتبر است.", current\_step)

else:

logger.warning("[ویزارد-POST] فرم مرحله ('%s') نامعتبر است. خطاها: %s", current\_step, form.errors.as\_json())

is\_last\_step = current\_step == self.steps.last

if form.is\_valid() and is\_last\_step:

logger.info("[ویزارد-POST] فرم مرحله آخر ('%s') معتبر است. انتظار اجرای done()...", current\_step)

elif form.is\_valid():

logger.info("[ویزارد-POST] فرم مرحله ('%s') معتبر است. انتظار رفتن به مرحله بعد...", current\_step)

try:

response = super().post(\*args, \*\*kwargs)

logger.info(f"[ویزارد-POST] اجرای متد post والد تکمیل شد. کد وضعیت پاسخ: {response.status\_code}")

logger.debug(f"[ویزارد-POST] داده‌های ذخیره‌شده در storage برای مرحله {current\_step}: {self.storage.get\_step\_data(current\_step)}")

if response.status\_code in (301, 302, 307, 308):

logger.info(f"[ویزارد-POST] ریدایرکت شناسایی شد. URL مقصد: {response.url}")

elif response.status\_code == 200:

logger.debug("[ویزارد-POST] پاسخ ۲۰۰ OK دریافت شد (یا خطای فرم یا بازگشت از done با خطا).")

return response

except Exception as e:

logger.error(f"[ویزارد-POST] خطا در حین اجرای متد post والد: {e}", exc\_info=True)

messages.error(self.request, \_("خطای پیش‌بینی نشده در پردازش مراحل ویزارد."))

return self.render\_goto\_step(self.steps.first)

۲. اصلاح فرم‌ست آیتم‌ها

برای اطمینان از رعایت شرط min\_num=1، یک فرم‌ست سفارشی تعریف شد:

python

from django.forms import inlineformset\_factory, BaseInlineFormSet

from django.core.exceptions import ValidationError

import logging

logger = logging.getLogger(\_\_name\_\_)

class WizardFactorItemFormSet(BaseInlineFormSet):

def clean(self):

super().clean()

valid\_forms = [form for form in self.forms if form.is\_valid() and not form.cleaned\_data.get('DELETE', False)]

logger.debug(f"Number of valid forms: {len(valid\_forms)}")

if len(valid\_forms) < self.min\_num:

raise ValidationError(\_('حداقل یک ردیف معتبر مورد نیاز است.'))

WizardFactorItemFormSet = inlineformset\_factory(

parent\_model=Factor,

model=FactorItem,

form=WizardFactorItemForm,

fields=('description', 'quantity', 'unit\_price'),

formset=WizardFactorItemFormSet,

extra=1,

can\_delete=True,

min\_num=1,

validate\_min=True,

)

۳. بهبود نمایش خطاها در قالب

برای نمایش خطاهای فرم‌ست در قالب step\_factor\_items.html، کد زیر پیشنهاد شد:

html

{% if formset.errors or formset.non\_form\_errors %}

<div class="alert alert-danger alert-sm p-2" role="alert">

<strong>{% trans "خطا در آیتم‌های فاکتور:" %}</strong><br>

{% for error in formset.non\_form\_errors %}

{{ error }}<br>

{% endfor %}

{% for form in formset %}

{% for field, errors in form.errors.items %}

{% for error in errors %}

{{ field.label }}: {{ error }}<br>

{% endfor %}

{% endfor %}

{% endfor %}

</div>

{% endif %}

۴. بررسی تنظیمات instance

متد get\_form\_instance بررسی شد تا اطمینان حاصل شود که فرم‌ست به درستی به مدل Factor متصل است. پیشنهاد شد که یک نمونه موقت Factor ایجاد شود:

python

def get\_form\_instance(self, step):

logger.debug(f"[Wizard-get\_form\_instance] دریافت instance برای مرحله: {step}")

if step == 'factor\_items':

logger.debug(f"[Wizard-get\_form\_instance] مرحله '{step}' فرم‌ست است، instance موقت ایجاد می‌شود.")

return Factor() # نمونه موقت بدون ذخیره

return super().get\_form\_instance(step)

۵. اقدامات دیباگ

بررسی درخواست‌های شبکه در تب Network مرورگر.

غیرفعال کردن موقت اعتبارسنجی برای تست.

افزودن لاگ‌های بیشتر برای بررسی داده‌های ذخیره‌شده در storage.

طراحی ساختار سازمانی

هدف

هدف طراحی یک ساختار سلسله‌مراتبی است که شامل:

سازمان: بالاترین سطح (مثلاً دفتر مرکزی یا مجتمع).

معاونت: زیرمجموعه سازمان (مثلاً معاونت مالی، عملیاتی).

اداره: زیرمجموعه معاونت (مثلاً اداره حسابداری).

شعبه: زیرمجموعه اداره (مثلاً شعبه منطقه ۱).

پست‌ها در هر سطح با امکان تعریف نقش‌های تأیید/رد کننده برای فرآیندهای خاص (مثلاً فاکتور، بودجه).

مدل‌های پیشنهادی

۱. مدل OrganizationalUnit (واحد سازمانی)

برای تعریف معاونت‌ها، ادارات، و شعب:

python

class OrganizationalUnit(models.Model):

UNIT\_TYPES = (

('DEPUTY', \_('معاونت')),

('DEPARTMENT', \_('اداره')),

('BRANCH', \_('شعبه')),

)

name = models.CharField(max\_length=100, verbose\_name=\_("نام واحد"))

code = models.CharField(max\_length=10, verbose\_name=\_("کد واحد"))

unit\_type = models.CharField(max\_length=10, choices=UNIT\_TYPES, verbose\_name=\_("نوع واحد"))

organization = models.ForeignKey(

'Organization',

on\_delete=models.CASCADE,

related\_name='units',

verbose\_name=\_("سازمان")

)

parent = models.ForeignKey(

'self',

null=True,

blank=True,

on\_delete=models.SET\_NULL,

related\_name='children',

verbose\_name=\_("واحد والد")

)

description = models.TextField(blank=True, null=True, verbose\_name=\_("توضیحات"))

is\_active = models.BooleanField(default=True, verbose\_name=\_("فعال"))

def \_\_str\_\_(self):

return f"{self.name} ({self.unit\_type}) - {self.organization.code}"

def clean(self):

from django.core.exceptions import ValidationError

if self.unit\_type == 'DEPUTY' and self.parent and self.parent.unit\_type != 'ORGANIZATION':

raise ValidationError(\_('معاونت فقط می‌تواند زیرمجموعه سازمان باشد.'))

if self.unit\_type == 'DEPARTMENT' and self.parent and self.parent.unit\_type != 'DEPUTY':

raise ValidationError(\_('اداره باید زیرمجموعه معاونت باشد.'))

if self.unit\_type == 'BRANCH' and self.parent and self.parent.unit\_type != 'DEPARTMENT':

raise ValidationError(\_('شعبه باید زیرمجموعه اداره باشد.'))

class Meta:

verbose\_name = \_("واحد سازمانی")

verbose\_name\_plural = \_("واحدهای سازمانی")

unique\_together = [['organization', 'code', 'unit\_type']]

permissions = [

('OrganizationalUnit\_add', 'افزودن واحد سازمانی'),

('OrganizationalUnit\_update', 'بروزرسانی واحد سازمانی'),

('OrganizationalUnit\_view', 'نمایش واحد سازمانی'),

('OrganizationalUnit\_delete', 'حذف واحد سازمانی'),

]

۲. مدل Organization

برای تعریف سازمان‌ها (دفتر مرکزی، مجتمع‌ها):

python

class Organization(models.Model):

code = models.CharField(max\_length=10, unique=True, verbose\_name=\_("کد سازمان"))

name = models.CharField(max\_length=100, verbose\_name=\_("نام سازمان"))

org\_type = models.ForeignKey(

'OrganizationType',

on\_delete=models.SET\_NULL,

null=True,

blank=True,

verbose\_name=\_("نوع سازمان"),

related\_name='organizations'

)

description = models.TextField(blank=True, null=True, verbose\_name=\_("توضیحات"))

parent\_organization = models.ForeignKey(

'self',

on\_delete=models.SET\_NULL,

null=True,

blank=True,

verbose\_name=\_("سازمان والد")

)

is\_active = models.BooleanField(default=True, verbose\_name=\_("فعال"))

is\_core = models.BooleanField(default=False, verbose\_name=\_("دفتر مرکزی سازمان"))

def clean(self):

from django.core.exceptions import ValidationError

if self.is\_core and self.parent\_organization:

raise ValidationError(\_('دفتر مرکزی نمی‌تواند سازمان والد داشته باشد.'))

if self.is\_core:

existing\_core = Organization.objects.filter(is\_core=True, is\_active=True).exclude(pk=self.pk)

if existing\_core.exists():

raise ValidationError(\_('فقط یک سازمان می‌تواند به‌عنوان دفتر مرکزی فعال باشد.'))

def \_\_str\_\_(self):

org\_type\_str = self.org\_type.fname if self.org\_type else \_("نامشخص")

return f"{self.code} - {self.name} ({org\_type\_str})"

@property

def budget\_allocations(self):

from budgets.models import BudgetAllocation

return BudgetAllocation.objects.filter(organization=self)

def save(self, \*args, \*\*kwargs):

self.full\_clean()

super().save(\*args, \*\*kwargs)

class Meta:

verbose\_name = \_("سازمان")

verbose\_name\_plural = \_("سازمان‌ها")

default\_permissions = ''

permissions = [

('Organization\_add', 'افزودن سازمان'),

('Organization\_update', 'بروزرسانی سازمان'),

('Organization\_delete', 'حذف سازمان'),

('Organization\_view', 'نمایش سازمان'),

]

indexes = [

models.Index(fields=['code', 'org\_type']),

]

۳. مدل Post

برای تعریف پست‌ها با نقش‌های تأیید/رد کننده:

python

class Post(models.Model):

name = models.CharField(max\_length=100, verbose\_name=\_("نام پست"))

organization = models.ForeignKey(

Organization,

on\_delete=models.CASCADE,

verbose\_name=\_("سازمان"),

related\_name='posts'

)

unit = models.ForeignKey(

OrganizationalUnit,

null=True,

blank=True,

on\_delete=models.SET\_NULL,

verbose\_name=\_("واحد سازمانی"),

related\_name='posts'

)

parent = models.ForeignKey(

'self',

null=True,

blank=True,

on\_delete=models.SET\_NULL,

verbose\_name=\_("پست والد")

)

level = models.IntegerField(default=1, verbose\_name=\_("سطح"))

description = models.TextField(blank=True, null=True, verbose\_name=\_("توضیحات"))

is\_active = models.BooleanField(default=True, verbose\_name=\_("وضعیت فعال"))

max\_change\_level = models.IntegerField(default=1, verbose\_name=\_("حداکثر سطح تغییر"))

is\_payment\_order\_signer = models.BooleanField(default=False, verbose\_name=\_("مجاز به امضای دستور پرداخت"))

is\_approver = models.BooleanField(default=False, verbose\_name=\_("تأییدکننده"))

is\_rejector = models.BooleanField(default=False, verbose\_name=\_("ردکننده"))

def \_\_str\_\_(self):

unit = self.unit.name if self.unit else \_("بدون واحد")

return f"{self.name} ({self.organization.code}) - {unit}"

def clean(self):

from django.core.exceptions import ValidationError

if self.unit and self.unit.organization != self.organization:

raise ValidationError(\_('واحد سازمانی باید متعلق به سازمان پست باشد.'))

if self.parent and self.unit != self.parent.unit:

raise ValidationError(\_('پست والد باید در همان واحد سازمانی باشد.'))

def save(self, \*args, \*\*kwargs):

if self.parent:

self.level = self.parent.level + 1

else:

self.level = 1

if self.max\_change\_level < self.level:

self.max\_change\_level = self.level

super().save(\*args, \*\*kwargs)

class Meta:

verbose\_name = \_("پست سازمانی")

verbose\_name\_plural = \_("پست‌های سازمانی")

default\_permissions = ''

permissions = [

('Post\_add', 'افزودن پست سازمانی'),

('Post\_update', 'بروزرسانی پست سازمانی'),

('Post\_view', 'نمایش پست سازمانی'),

('Post\_delete', 'حذف پست سازمانی'),

]

۴. مدل ApprovalRole (اختیاری)

برای مدیریت نقش‌های تأیید/رد کننده با جزئیات بیشتر:

python

class ApprovalRole(models.Model):

PROCESS\_TYPES = (

('FACTOR', \_('فاکتور')),

('BUDGET', \_('بودجه')),

('DOCUMENT', \_('اسناد')),

)

post = models.ForeignKey(

Post,

on\_delete=models.CASCADE,

related\_name='approval\_roles',

verbose\_name=\_("پست")

)

process\_type = models.CharField(max\_length=20, choices=PROCESS\_TYPES, verbose\_name=\_("نوع فرآیند"))

can\_approve = models.BooleanField(default=False, verbose\_name=\_("می‌تواند تأیید کند"))

can\_reject = models.BooleanField(default=False, verbose\_name=\_("می‌تواند رد کند"))

stage\_order = models.IntegerField(default=1, verbose\_name=\_("ترتیب مرحله تأیید"))

def \_\_str\_\_(self):

return f"{self.post.name} - {self.get\_process\_type\_display()} (Approve: {self.can\_approve}, Reject: {self.can\_reject})"

class Meta:

verbose\_name = \_("نقش تأیید")

verbose\_name\_plural = \_("نقش‌های تأیید")

unique\_together = [['post', 'process\_type', 'stage\_order']]

permissions = [

('ApprovalRole\_add', 'افزودن نقش تأیید'),

('ApprovalRole\_update', 'بروزرسانی نقش تأیید'),

('ApprovalRole\_view', 'نمایش نقش تأیید'),

('ApprovalRole\_delete', 'حذف نقش تأیید'),

]

۵. مدل OrganizationType

بدون تغییر:

python

class OrganizationType(models.Model):

fname = models.CharField(max\_length=100, unique=True, null=True, blank=True, verbose\_name=\_('نام شعبه/مجتمع/اداره'))

org\_type = models.CharField(max\_length=100, unique=True, null=True, blank=True, verbose\_name=\_('نوع شعبه/مجتمع/اداره'))

is\_budget\_allocatable = models.BooleanField(default=False, verbose\_name=\_("قابل استفاده برای تخصیص بودجه"))

def \_\_str\_\_(self):

return f"{self.fname} - {self.org\_type} " or \_("نامشخص")

class Meta:

verbose\_name = \_('عنوان مرکز/شعبه/اداره/سازمان')

verbose\_name\_plural = \_('عنوان مرکز/شعبه/اداره/سازمان')

default\_permissions = ''

permissions = [

('OrganizationType\_add', 'افزودن شعبه/اداره/مجتمع/سازمان'),

('OrganizationType\_view', 'نمایش شعبه/اداره/مجتمع/سازمان'),

('OrganizationType\_update', 'ویرایش شعبه/اداره/مجتمع/سازمان'),

('OrganizationType\_delete', 'حذف شعبه/اداره/مجتمع/سازمان'),

]

مهاجرت داده‌ها

برای انتقال داده‌های موجود از فیلد branch در مدل Post به OrganizationalUnit:

python

def migrate\_branches\_to\_units(apps, schema\_editor):

OrganizationalUnit = apps.get\_model('your\_app', 'OrganizationalUnit')

Post = apps.get\_model('your\_app', 'Post')

branch\_mapping = {

'OPS': ('بهره‌برداری', 'DEPUTY'),

'FIN': ('مالی و اداری', 'DEPUTY'),

None: ('بدون واحد', 'DEPUTY'),

}

for post in Post.objects.all():

branch\_name, unit\_type = branch\_mapping.get(post.branch, ('بدون واحد', 'DEPUTY'))

unit, created = OrganizationalUnit.objects.get\_or\_create(

organization=post.organization,

unit\_type=unit\_type,

code=post.branch or 'NONE',

defaults={'name': branch\_name, 'is\_active': True}

)

post.unit = unit

post.branch = None

post.save()

class Migration(migrations.Migration):

dependencies = [...]

operations = [

migrations.RunPython(migrate\_branches\_to\_units),

migrations.RemoveField(model\_name='Post', name='branch'),

]

فرم‌ها و ویوها

فرم برای Post

برای محدود کردن واحدهای سازمانی به سازمان انتخاب‌شده:

python

class PostForm(forms.ModelForm):

class Meta:

model = Post

fields = ['name', 'organization', 'unit', 'parent', 'description', 'is\_active', 'max\_change\_level', 'is\_payment\_order\_signer', 'is\_approver', 'is\_rejector']

def \_\_init\_\_(self, \*args, \*\*kwargs):

super().\_\_init\_\_(\*args, \*\*kwargs)

if 'organization' in self.data:

try:

org\_id = int(self.data.get('organization'))

self.fields['unit'].queryset = OrganizationalUnit.objects.filter(organization\_id=org\_id)

except (ValueError, TypeError):

self.fields['unit'].queryset = OrganizationalUnit.objects.none()

elif self.instance and self.instance.organization:

self.fields['unit'].queryset = OrganizationalUnit.objects.filter(organization=self.instance.organization)

def clean(self):

cleaned\_data = super().clean()

unit = cleaned\_data.get('unit')

organization = cleaned\_data.get('organization')

if unit and organization and unit.organization != organization:

raise forms.ValidationError(\_('واحد سازمانی باید متعلق به سازمان انتخاب‌شده باشد.'))

return cleaned\_data

منطق تأیید در ویو

برای مدیریت فرآیند تأیید فاکتور:

python

def approve\_factor(request, factor\_id):

factor = get\_object\_or\_404(Factor, id=factor\_id)

user\_post = Post.objects.filter(user=request.user).first()

if user\_post and user\_post.is\_approver and factor.organization == user\_post.organization:

factor.status = 'APPROVED'

factor.save()

messages.success(request, \_('فاکتور تأیید شد.'))

else:

messages.error(request, \_('شما اجازه تأیید این فاکتور را ندارید.'))

return redirect('factor\_detail', factor\_id=factor.id)

ملاحظات و نکات

ویزارد فاکتور

دیباگ: بررسی لاگ‌ها و درخواست‌های شبکه برای شناسایی خطاهای پنهان.

تست: تست دستی با افزودن ردیف‌های معتبر و غیرفعال کردن موقت اعتبارسنجی.

رابط کاربری: اطمینان از نمایش صحیح خطاها در قالب.

ساختار سازمانی

انعطاف‌پذیری: مدل OrganizationalUnit امکان افزودن سطوح جدید را فراهم می‌کند.

مهاجرت داده‌ها: مهاجرت باید با دقت انجام شود تا داده‌های موجود حفظ شوند.

عملکرد: افزودن ایندکس‌ها برای بهبود عملکرد کوئری‌ها.

امنیت: محدود کردن دسترسی کاربران به پست‌ها و واحدهای سازمانی مربوط به سازمان خود.

رابط کاربری: استفاده از فیلترهای پویا در فرم‌ها برای بهبود تجربه کاربری.

نحوه تولید فایل ورد

متن بالا را کپی کنید.

آن را در Microsoft Word جای‌گذاری کنید.

ساختار را با سرفصل‌ها، فونت، و فاصله‌گذاری مناسب تنظیم کنید.

فایل را به صورت .docx ذخیره کنید.

اگر نیاز به کمک در تولید فایل ورد یا تنظیمات خاص دارید، اطلاع دهید تا راهنمایی بیشتری ارائه کنم!